Gramaglia, Marco, et al. (2011). Performance evaluation of a tree-based routing and address autoconfiguration for vehicle-to-Internet communications. *2011 11th International Conference on ITS Telecommunications (ITST 2011), St. Petesburg, Russia, 23-25 August 2011. Proceedings.* Pp. 45-50. USA: IEE, cop. 2011

DOI: https://doi.org/10.1109/ITST.2011.6060101

Performance evaluation of a Tree-Based Routing and Address Autoconfiguration for Vehicle-to-Internet Communications

Marco Gramaglia*†, Carlos J. Bernardos†, Maria Calderon†, Antonio de la Oliva†
* Institute IMDEA Networks, Avda. del Mar Mediterraneo 22, 28918 Madrid, SPAIN
E-mail: marco.gramaglia@imdea.org
† Universidad Carlos III de Madrid, Avda. Universidad 30, 28911 Leganés, SPAIN
E-mail: {cjbc, maria, aoliva}@it.uc3m.es

***Abstract*—Vehicular ad hoc networks have proven to be quite useful for broadcast alike communications between nearby cars, but can also be used to provide Internet connectivity from vehicles. In order to do so, vehicle-to-Internet routing and IP address autoconfiguration are two critical pieces. TREBOL is a tree-based and configurable protocol which benefits from the inherent tree-shaped nature of vehicle to Internet traffic to reduce the signaling overhead while dealing efficiently with the vehicular dynamics. This paper experimentally evaluates the performance of TREBOL using a Linux implementation under lab-controlled realistic scenarios, including real vehicular traces obtained in the region of Madrid.**

I. Introduction

Enabling vehicles to *talk* – that is, to be able to exchange useful information that can be used for example to prevent traffic accidents or let drivers know of critical traffic conditions ahead – has been and continue to be a key research topic. Most of the work has been focused so far on human safety and traffic efficiency applications, which mostly make use of broadcast communications. However, drivers and passengers also demand to enjoy Internet connectivity from their vehicles, so classical and new Internet applications could be enabled in cars. Providing vehicles with Internet connectivity would additionally help speeding up the adoption of vehicular communication systems by the users, since they will see an additional benefit in the installation of communication systems in their cars.

The use of vehicular ad hoc networks (VANETs) is the most widely adopted approach to provide connectivity in vehicular environments in a cheap and scalable way. A VANET is an ad hoc, multi-hop and short-range wireless network, formed by vehicles in a certain area and fixed roadside gateways placed along the roads. Compared to other wireless communication approaches, using a multi-hop solution brings benefits to the user (i.e., cost savings and high bandwidth), and to the network providers that can alleviate their already overloaded 3G infrastructure. In real deployments, these roadside gateways (RSGs) can be co-located with the Road Side Units (RSUs) deployed around the roads for safety purposes.

To enable Vehicle-to-Internet communications, some functionalities are needed:

- *Address configuration:* Vehicles have to be able to autoconfigure a valid IP network address in an automatic way, without requiring manual intervention from the user.
- *Routing capability:* mechanisms for an efficient routing of IP datagrams, mainly unicast, from the vehicle to roadside gateways and vice versa.
- *Mobility management:* vehicular networks are characterized by high mobility. Thus, an effective mechanism for seamless handover between different networks and roadside gateways is required.

We proposed TREBOL in [1], a tree-based protocol for vehicle-to-Internet communications, which addresses two of the previously mentioned functionalities, namely routing and address configuration. In this paper we experimentally validate TREBOL, by implementing the protocol in a controlled testbed, and performing tests using real traffic traces in an emulated vehicular environment.

The rest of the paper is organized as follows. Section II introduces how TREBOL works, while in Section III we describe our prototype implementation of TREBOL. We report our experimental evaluation of TREBOL in Section IV, and Section V finalizes this paper.

II. Trebol

TREBOL [1] is a tree-based routing and IP address autoconfiguration protocol which bases its data forwarding decisions on IPv6 addresses (i.e., it is a topological routing protocol). Data paths follow a tree built by the TREBOL protocol, which is formed using position information (e.g., vehicles are assumed to have a GPS receiver) to minimize the control overhead load. We describe next how this is achieved. Let's focus first on the routing capabilities of TREBOL, by assuming for the time being that nodes are already provided with IPv6 addresses.

We argue that vehicle-to-Internet unicast communications exhibit a common set of characteristics that may be exploited by the VANET routing protocol. In particular, not all network nodes behave in the same way: roadside gateways (RSG) play a critical role, since they operate as relays to the Internet. The required network connectivity graph is anchored at the RSG (i.e., all data traffic traverses the RSG), as opposed to other

978-1-61284-671-2/11/$26.00

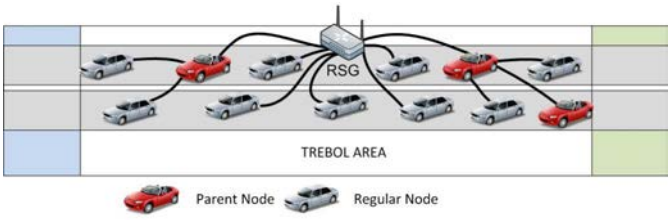


Fig. 1. TREBOL area

vehicle scenarios, in which a mesh graph is desired. On the other hand, as opposed to the very popular geographical routing protocols, TREBOL forwarding is not based on positions, so neither beacon messages nor location service information is needed, allowing great savings in terms of control overhead.

In TREBOL, the **upstream tree** (i.e., the tree used in the forwarding of data packets from the vehicle to the Internet) is built and updated when each node learns about its parent upon receiving periodical conf guration messages (CM) sent by the roadside gateway (RSG). It is assumed that each RSG plays the role of relay (i.e., forwarding traff c from/to the Internet) for the vehicles within a limited geographical area, known as TREBOL area (see Fig. 1). Thus, conf guration messages sent by a RSG are spread within its TREBOL area. On the other hand, the creation of the downstream tree (i.e., the tree used in the forwarding of data packets from the Internet to the vehicle) follows a reactive approach: each node learns who are its children on a per data packet basis, as part of the forwarding of data packets.

As already mentioned, TREBOL builds and refreshes the upstream tree by using periodical conf guration messages (identif ed by a unique and incremental *sequence number*) which are initially sent by the RSG and then regenerated and sent by a subset of the VANET nodes. Once a node receives a CM with a newer sequence number, the sender of that CM becomes the parent of the receiving node, and the forwarding state is updated accordingly (i.e., the parent is used as next hop for upstream data traff c towards the Internet). Then, the node regenerates the CM (i.e., updating some f elds but keeping the original sequence number) and sets a backoff timer. Only if this backoff timer expires, the node broadcasts this regenerated CM to its neighbors. In the meantime, if the node receives another CM with this same sequence number (i.e., sent by another node with a shorter backoff time), it cancels the sending of the regenerated CM. It is worth mentioning that only if a node sends a regenerated CM, it has the chance to become a *parent* node. *Parent* nodes take the responsibility of forwarding data traff c from/to the Internet from/to its descendants, so a critical issue in TREBOL is to select as parents those nodes that according to their characteristics (e.g., speed, position, etc.) lead to more stable trees. The CMs sent by the RSG include the following information:

- $areaBoundary$: geographic information describing the TREBOL area. Nodes outside this area receiving a CM discard the message.
- $sendPos$: geographic position of the sender of the CM. It is set initially to the location of the RSG and then overwritten with the position of the last node that regenerated and sent the CM.
- $prefR$: value that represents the preferred distance between consecutive *parents* (i.e., nodes with children). Lower values imply more dense, populated trees, while higher ones imply sparse trees.
- R: value f xing the maximum allowed distance between the receiver and the sender (i.e. the RSG or a potential parent node) of the CM. If the sender is farther away from the receiver node than R (i.e., $sendPos$ fi ld), then the CM is discarded. In this way, R serves as a virtual wireless coverage radius.
- $prefS$: value that represents the preferred speed of nodes sending regenerated CMs (i.e., potential *parent* nodes). It is set by the RSG. This value is used to preserve the stability of the tree selecting as *parent* nodes those that travel at similar speeds (closer to $prefS$).
- $maxSpeedDiff$: nodes whose speed differs more than this value from $prefS$ will be prevented from sending regenerated CMs (i.e., becoming *parent* nodes).
- D_{pos} and D_{speed}: these two values set the maximum value for the backoff timer. The higher these values are, the more time is required to build the tree. On the other hand, too short values might cause many wireless collisions.

Selecting the potential *parent* nodes is a completely distributed process based on a backoff timer:

$$T_{backoff} = \frac{\|((\|pos - sendPos\|) - prefR)\|}{R} \times D_{pos} + \frac{\|speed - prefS\|}{maxSpeedDiff} \times D_{speed}$$

where pos is the node's position and $speed$ is the node's speed.

A node that is located at a distance $prefR$ from the sender of the CM, and that travels at a speed of $PrefSpeed$ would immediately send the regenerated CM ($T_{backoff} = 0\ s$). After waiting $T_{backoff}$ seconds, the node sends the regenerated CM (with the $sendPos$ fi ld updated) only if it has not received another CM with the same sequence number from one of its neighbors before. In this way, the shorter the $T_{backoff}$ of a node is, the more likely the node sends a regenerated CM becoming a potential *parent* (i.e., assuming the responsibility of having children and forwarding their data traff c).

On the other hand, the TREBOL **downstream tree** (i.e., the tree followed to deliver data traff c from the Internet to the vehicle) is built and refreshed on a per data packet basis as part of the data packets forwarding process. A node will be aware of the identity (i.e., the IPv6 address) of its descendants (i.e., downstream nodes in the tree) when it receives data traff c addressed to the Internet from one of its children (i.e., the child has selected the node as next hop for traff c towards the Internet). Thus, upon receiving a data packet addressed to the Internet, the node learns the identity of the descendant

(i.e., the source address of the data packet) and updates the corresponding forwarding state information (i.e., the child which forwarded this data packet becomes the next hop for downstream data traffic towards the descendant).

So far we have assumed that VANET nodes are already provided with an IP address that can be used by the TREBOL routing mechanism as identifier in the forwarding process. The same CM messages used for building up the upstream tree could also be used to convey IPv6 prefix information, allowing nodes to autoconfigure IP addresses in a way similar to the standard IPv6 SLAAC [2], as described next. All nodes within the same TREBOL area share the same IPv6 prefix (or set of prefixes), effectively forming a multi-link subnet. The RSG sends standard Router Advertisements (RAs) messages, modified as follows by TREBOL: *i)* RAs are regenerated by each *parent* node, keeping the same prefix, and *ii)* RAs are used by all VANET nodes (including *parent* nodes, which are also routers) to autoconfigure an address from the prefix. These RAs are extended with additional options to carry the fields defined in the CMs (needed by TREBOL routing). In order to avoid unnecessary control overhead, Duplicate Address Detection (DAD) is disabled.

This approach reduces the overall control overhead required by combining routing and address autoconfiguration functions using a single set of signaling messages [1].

III. Implementation

Number and high mobility of nodes (i.e., vehicles) and lack of connectivity due to sparse traffic (i.e. low vehicle density) are some of the challenges that VANETs have to face to, in comparison to more traditional mobile ad hoc networks (MANETs). These problems, together with other low-layer related issues (e.g., severe wireless conditions at high speed and obstacles), make it very difficult to conduct realistic experiments. Some testbeds for VANET applications have been deployed so far, among them we highlight the following: Cartel and Cabernet projects at MIT [3], [4], Dome and DieselNet at Amherst [5], VanLan by Microsoft Research [6] and C-VeT at UCLA [7].

Our long-term goal is to develop a low-cost, flexible VANET emulation platform using COTS wireless devices to get more insight of some aspects of VANET protocols that cannot be investigated using a simulator (tool already used to evaluate TREBOL in [1]), as for example implementation complexity or behaviour of the protocol with real data traffic. This kind of testbed does not aim at looking for a realistic emulation of the wireless medium, but we rather focus on how the protocol reacts using a multi-hop connectivity map based on nodes position and movement. Studying the behaviour of a VANET protocol using an emulated environment fed by real traffic traces and realistic mobility patterns can help to analyze how the protocol operates under conditions that cannot be easily reproduced in a simulator. We plan to use as COTS devices, the well-known wireless SOHO router Linksys WRT54GL [1], but in this paper we report on first validation

[1]http://www.linksysbycisco.com/EU/en/products/WRT54GL

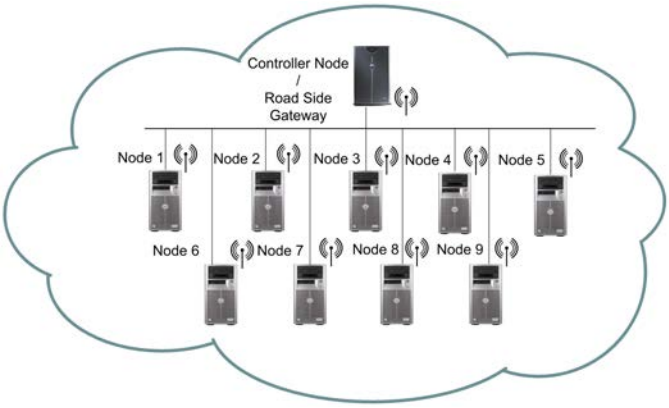


Fig. 2. Deployed testbed

experiences using a smaller PC-based testbed consisting in 10 PCs running a Linux system, and TREBOL as VANET protocol to be analyzed. The installed OS distribution is Ubuntu 10.10 running a 2.6.35 kernel. Each node is equipped with an Atheros AR5001X wireless card managed by the `ath5k` driver. We are currently working on the migration of this PC-based testbed to the Linksys-based one.

Fig. 2 depicts the architecture of the testbed deployed in a laboratory of the Department of Telematics Engineering of the Universidad Carlos III de Madrid. All nodes in the testbed are configured in `ad-hoc` mode, belong to the same IBSS, and therefore are configured on the same IEEE 802.11a channel (i.e., 140). Nodes are placed in a reduced space, and consequently can all directly communicate with each other (as they are all within 1-hop radio coverage), thus creating a full-mesh topology. Note that an Ethernet network is used for controlling and result-gathering purposes.

In order to emulate a dynamic multi-hop connectivity map of the nodes traveling in a road, it is necessary to know the complete route of each vehicle in the considered stretch, a feature provided by the SUMO [2] microscopic traffic simulator. SUMO supports many mechanisms for providing the input traffic rate of the system, but our choice was to use real vehicular traces kindly supplied by the Madrid city council. The measurements, collected at a fixed observation point along the M-30[3] orbital motorway, provide a time mark and the sensed speed for each vehicle that goes through the checkpoint. Feeding the simulator with real traces which have a resolution of 0.1 seconds gives a good estimation of the nodes position at any time. The final output provided by SUMO is a trace file for each vehicle providing the vehicle's position and speed at every time step. Using these traces we can calculate the connectivity map for each node at any time. For this evaluation we used the unit disk coverage rule (with the parameter coverage radius, R) but more complex rules can be used, taking into account the vehicles relative speed or the presence of obstacles. The outcome of this procedure is to obtain, for

[2]http://sumo.sourceforge.net/

[3]http://en.wikipedia.org/wiki/M30_motorway

each node, the connectivity map at any moment.

In our testbed all nodes are within 1-hop direct radio coverage. To emulate a multi-hop connectivity environment we artificially inhibit the wireless connections using a software module. We implemented a library that, using the `ip6tables` Kernel API [4], can emulate a dynamically changing multi-hop wireless connectivity graph. As each vehicle is bound to a single machine the connectivity mapping can be represented using the wireless card MAC addresses. Hence, at each time step, the firewall rules are updated allowing traffic coming only from the *neighbor* wireless cards and, thus, creating an emulated virtual topology over the full-mesh real one.

This approach requires time synchronization among nodes, a task that we accomplished running a pacemaker module in the controller node. The synchronization is kept by the reception of broadcast time-step messages on the Ethernet control network. By merging the time-step information with the generated vehicle trace, each node can create a snapshot of its current connectivity map. The TREBOL software client is in charge of processing the time-step messages and updating position, speed and neighbors set (using the aforementioned library) for each node. The positioning information is also used by TREBOL in order to calculate the backoff timer while processing a new RA. TREBOL has to change the forwarding table in two situations: during the tree-refreshing phases and when the node is forwarding data on behalf of a child node. In the first case, the RA source address (i.e., the parent node address) is taken as the default gateway to the Internet. In the latter case TREBOL, using the `libpcap`[5] API, gets the source address of the data packets it is relaying and updates the routing table accordingly. `Netlink` sockets [8] are used for both of the tasks. Finally, the software keeps track of all the routes added for the downstream tree and periodically cleans the forwarding table, removing all the unused entries.

IV. Results

In our previous work [1], we focused on comparing TREBOL by simulation with a geographic based routing protocol. In this paper, we aim at evaluating the behavior of TREBOL in a more realistic environment, looking at how the algorithm reacts to possible wireless malfunctions and how real data traffic requirements are met by the emulated moving network. We have selected three metrics: *i)* the parent nodes placement, *ii)* the total tree construction time, and, *iii)* the TCP throughput.

Regarding the first of these metrics, analysis of the parent locations selected by TREBOL, this metric can provide insights on how the protocol reacts to configuration parameters changes[6]. Moreover, it can also prove the resilience of the algorithm to losses of Router Advertisements. We set up a emulation scenario representing a stretch of road 2Km long, with an RSG placed half way, and a total of 9 cars entering the road (at position 0) with times and speeds reflecting the actual ones in a real scenario as explained in Section III. The average distance between the first and the last vehicle of the queue is around 500m. The test is run 40 times leading to about 240 tree-refreshing phase (the time between consecutive RAs, T_{RA}, is uniformly distributed between 1.5s and 2.5s). Results are shown in Fig. 3. In the sub-figures we can see the effect of varying one of the TREBOL configuration parameters, $prefR$ (which influences on the desired distance between consecutive parents): 120m for Fig. 3(a), 150m for Fig. 3(b) and 180m for Fig. 3(c) keeping the coverage radius R at 200m. We can see that the protocol behaves as expected: using a $prefR$ value close to the maximum coverage radius forces parent nodes to be more separated between them, while shorter values make the parents topology much denser.

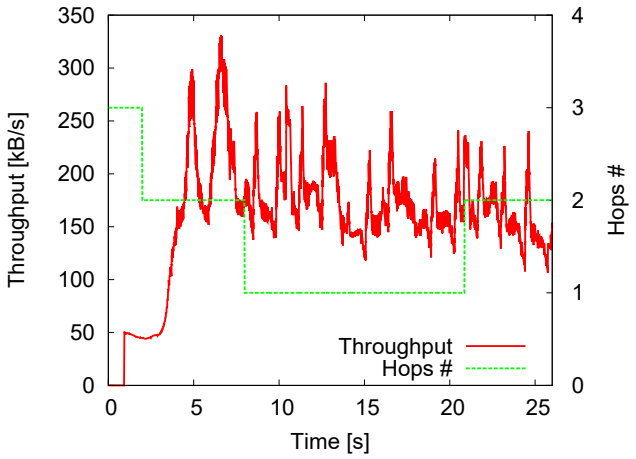


Fig. 5. UDP Throughput and the hop distance from the RSG

TABLE I
Tree construction times

$prefR$	T_{build} [s]	Min [s]	Max [s]
180	0.087	0.022	0.214
150	0.088	0.025	0.21
120	0.09	0.03	0.215

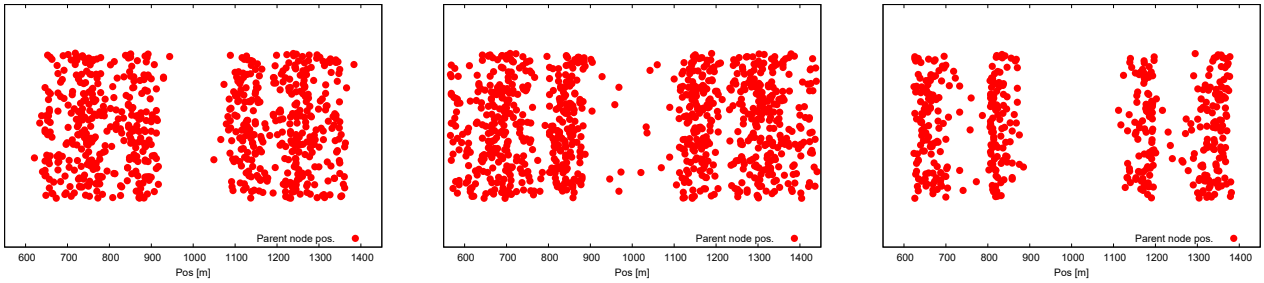
Another important aspect is the time required to build the tree. This time depends on the real topology (well placed nodes have shorter backoff times) and on the configuration parameters. The number of chosen parents (value influenced by $prefR$) might have an impact on this time: the higher the number of the hops, the higher the tree construction time. Choosing too short values for the parameters that have an impact on the backoff timer (see [1] for details) might cause problems for parent selection, as retransmission attempts would be scheduled too close. On the other hand higher values will increase the tree construction time, worsening the effect of asymmetrical paths. Notice that during tree-refreshing phase the downstream and upstream tree may not match exactly for short periods of time, which may cause asymmetrical paths.

Results (with the same setup as the previous test) are shown in Table I. With this setup, we fixed $D_{pos} = 0.04s$ and

[4]http://www.netfilter.org/projects/iptables/

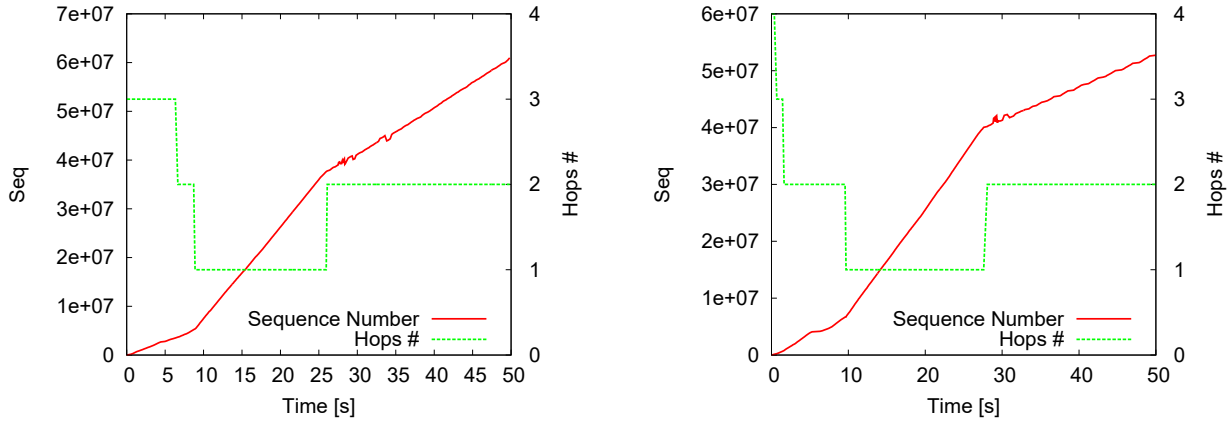
[5]http://www.tcpdump.org/

[6]We have not explained in detail how TREBOL works due to space limitation constraints. The behavior of TREBOL can be influenced by tuning some parameters [1].



(a) $prefR = 120m$ (b) $prefR = 150m$ (c) $prefR = 180m$

Fig. 3. Node Placement



(a) TCP Throughput run #1 (b) TCP Throughput run #2

Fig. 4. TCP Throughput and the hop distance from the RSG

$D_{speed} = 0.01s$. The tree construction time (T_{build}), covering 2 or 3 hops (according to Fig. 3), is on average around $0.9s$. This value includes the backoff procedure, the wireless medium access time and the packet processing time for each hop. Also the minimum and the maximum $\bar{T}_{build}$ does not seem to be significantly affected by $prefR$, but only by D_{pos} and D_{speed}.

A second goal of this paper is to show how a network running TREBOL performs under real data traffic conditions . The first test we run consists in streaming a video [7] from a server at the infrastructure to a vehicle. For this purpose we used RTP/RTCP [9] for delivering the multimedia flow. The request for the video flow is done using the RTSP [10] protocol.

Fig. 5 shows that the throughput consumed by the video is not influenced by the mechanism used by TREBOL to update the routes. Although the bitrate value varies over the time (the video is encoded using Ogg/Vorbis [11], [12], a VBR codec), the average value is kept stable with the number of hops.

Finally, we evaluated the performance of a TCP connection (namely a HTTP session) to check the effect that asymmetrical paths may have on the offered throughput. As already mentioned, the TREBOL tree refresh process can lead to asymmetric paths, which are known to affect TCP performance [13]. Although available rate and RTT do not change dramatically in our testbed, we wanted to make sure that the TREBOL tree refresh process would not affect the TCP performance heavily. Our test was done using a node initially placed 400m far from the RSG. Once it is configured, the node starts downloading a huge text file stored in a Web Server running in the RSG, while the node keeps traveling along the road, first getting closer to the RSG (in terms of distance and hence of hop numbers) and then moving away.

Results in Fig. 4 show that the tree refresh process only affects the TCP connection while moving away from the RSG (i.e., when increasing the hop number). There, due to the increasing RTT value, the throughput oscillates for short periods. However, as shown in Fig. 4, TCP Cubic [14] (the used TCP favor, it comes by default in Linux kernels since the version 2.6.19) can easily manage this situation.

V. Conclusion

TREBOL is a tree-based routing and IP address autoconfiguration protocol, proposed in [1], that benefits from the inherent tree-shaped nature of vehicle-to-Internet traffic to reduce the signaling overhead while dealing efficiently with vehicular dynamics.

The performance of TREBOL was analyzed in [1] based on simulations. In this paper we have gone a step further, by

[7] http://www.sintel.org/

developing a real prototype of TREBOL in Linux. Besides, an emulation testbed has also been designed and deployed, allowing the simulation of dynamic multi-hop connectivity patterns (such as the ones found in real traffic situations) on top of a physical testbed deployment in which all nodes are within direct radio coverage. SUMO and real traffic traces from Madrid are used in the experiments, with the goal of emulating scenarios as close as possible to real ones. Obtained results do not only show that TREBOL can be implemented in real devices and works as expected, but also that the achieved performance is good enough for a broad range of applications, covering both UDP and TCP ones.

Next steps include the migration of the VANET emulation testbed to COTS devices as well as the analysis of how to better emulate wireless degradation conditions (not just hard connectivity decisions) within a reduced physical space.

VI. Acknowledgements

The authors would like to acknowledge the Madrid city council for kindly providing us with the vehicular traces used in this work.

The research of Marco Gramaglia and Carlos J. Bernardos leading to these results has been supported by the Ministry of Science and Innovation of Spain under the QUARTET project (TIN2009-13992-C02-01). The work of Marco Gramaglia, Carlos J. Bernardos and Antonio de la Oliva has also been supported by the European Community's Seventh Framework Programme (FP7-ICT-2009-5) under grant agreement n. 258053 (MEDIEVAL project).

References

[1] M. Gramaglia, M. Calderon, and C. J. Bernardos, "TREBOL: Tree-Based Routing and Address Autoconfiguration for Vehicle-to-Internet Communications," in *IEEE Vehicular Technology Conference (VTC)*, May 2011.

[2] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), September 2007.

[3] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. K. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: A distributed mobile sensor computing system," in *4th ACM SenSys*, Boulder, CO, November 2006.

[4] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular content delivery using wifi," in *14th ACM MOBICOM*, San Francisco, CA, September 2008.

[5] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan, "Interactive WiFi Connectivity for Moving Vehicles," in *Proc. ACM SIGCOMM*, August 2008.

[6] R. Mahajan, J. Zahorjan, and B. Zill, "Understanding wifi-based connectivity from moving vehicles," in *In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007.

[7] P. Lutterotti, G. Pau, D. Jiang, M. Gerla, and L. Delgrossi, "C-vet, the ucla vehicular testbed: An open platform for vehicular networking and urban sensing," in *International Conference on Wireless Access for Vehicular Environments (WAVE 2008)*, 2008.

[8] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "Linux Netlink as an IP Services Protocol," RFC 3549, July 2003.

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[10] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, April 1998.

[11] I. Goncalves, S. Pfeiffer, and C. Montgomery, "Ogg Media Types," RFC 3534, September 2008.

[12] L. Barbato, "RTP Payload Format for Vorbis Encoded Audio," RFC 5215, August 2008.

[13] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of asymmetry on tcp performance," in *Mobile Computing and Networking*, pp. 77–89.

[14] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, July 2008.